ISSN 0956-9979

APRIL 1999

VIRUS BULLETIN

THE INTERNATIONAL PUBLICATION ON COMPUTER VIRUS PREVENTION, RECOGNITION AND REMOVAL

IN THIS ISSUE:

• **Just who is Melissa?** Hours before this issue went to press, W97M/Melissa.A crashed countless email servers across the US. A preliminary glance at this virus will be followed next month by a full analysis. See p.3.

• **Ones to watch:** This month Péter Ször gets happy with Win32/Ska and Shane Coursen discusses the various vulnerabilities of *PowerPoint*. Virus news starts on p.6.

• **Keeping in touch:** South Korea's leading anti-virus developer and vendor writes a letter to *VB* about the state of his nation, on p.13.

• **Through the keyhole:** A new series gets under way in this issue. A Day in the Life charts a typical 24 hours spent with a corporate professional responsible for anti-virus implementation, starting on p.15.

CONTENTS

GUEST EDITORIAL

Let's Do the Timewarp Again

“Anti-virus will continue to come in from the cold...”

The temptation to start this type of piece with ‘It's been an interesting couple of years since I last wrote for VB’ is enormous. However, nothing has really changed. For someone who has spent those years labouring over an anti-virus product, this realization is fairly painful, but for users of the very same products, it should be all the more troubling.

Focus on spring 1997 and the rise of the macro virus. While no-one is denying that the numbers have gone up, that the techniques have been modified in certain (comparatively minor) ways and that occasionally new viruses that are in some way ‘interesting’ appear, the fact remains that the field of macro viruses is relatively unchanged. Today, *Windows 9x*-specific viruses are on the increase. Despite new techniques in this field, let us not forget that two years ago the first such viruses were already written. Anti-virus products were either already able to detect and disinfect them, or they were well on the way there. The long-predicted appearance of genuine ‘Internet-aware’ viruses is only just beginning – if there is any doubt about it, look at the prevalence of Win32/Ska, alias Happy99 (see this issue, p.6). DOS viruses continue their gradual descent into the realms of history – the only remarkable thing being how long it is clearly going to take for them to disappear. DOS is going to be with us for a long time yet, and it seems inevitable that its viruses will live on with it.

In the field of anti-virus companies, there have been more than a few changes – the sharks at the helm of the large American corporates have been gobbling up smaller companies at a great rate. These changes merely continue the trends of many years – takeovers and buyouts in this arena are certainly not new. The pace has been hectic of late, as large companies scrabble for the technology that they once knew how to create themselves (forgotten somewhere between being a small company and being a large company), but while there is space in the industry for a couple more significant takeovers fairly soon, this looks set to calm down for a while.

However, anti-virus technology is certainly the most unchanging of everything I have mentioned. The commodification of anti-virus has resulted in the conversion of the fundamental parts of anti-virus products into easily reusable components, ready for connection with other types of product. Anti-virus will continue to come in from the cold – it will stop being a discipline entirely unto itself, with its own rules and its own petty power games, and will instead move into the global security arena as ‘just another aspect of safe computing’.

The fundamental ways in which we, the anti-virus companies, attempt to deal with the virus threat have not changed – and they are just as inadequate as ever. Scientifically speaking, the so-called ‘solutions’ in use today utilize techniques that are living on borrowed time. The creeping bloat of detection components can only end with the products being too prone to error, and too slow to use. Fundamentally, detection components now (that claim to detect 40,000+ viruses) work the same way they did when they claimed to detect a few hundred.

Anti-virus customers now know what they want from a product. When using a file, they don't want to be asked unintelligible questions about unauthorized attempts to access parts of this strange and wonderful entity called ‘the registry’ or those weird things called ‘sectors’ that seem to inhabit the hard disk. Rightly or wrongly, they want a definitive answer from their anti-virus product – either ‘this file is clean’ or ‘this file is infected with the QwertyUio.P virus’. Only known-virus scanning can be that certain, and so (in spite of all its failings) we still have it.

Finally, there is the tricky topic of the virus authors. There seems to be no sign of them ceasing to write viruses – while some individuals and groups may decide that they have better things to do with their time, there are always others waiting in the wings to replace them. The allure of virus writing has not waned in the slightest. Viruses are still very much a part of the everyday world of computing, and that shows no sign of changing either.

Ian Whalley

NEWS

Melissa Messenger?

Late on 26 March a *Word 97* macro virus ravaged corporate email systems in the US. Named W97M/Melissa.A, this class-infector has a payload designed to ensure its rapid spread. On first infection, Melissa attempts to extract 50 addresses from each address book visible to the *Microsoft Outlook* and emails a copy of the current, infected document to those addresses. The effect is exacerbated in many corporate environments where *Outlook* is used as a client to the *Microsoft Exchange* email server, as server-based address lists often contain addresses which are mailing lists. The ensuing email load slowed or crashed many email servers. As this issue goes to press the damage is still being repaired, and being the first 'business day' after the scourge started, the full extent of its spread has yet to be determined. An analysis will feature next month ∎

VB'99 Sponsors

This year, an innovative scheme of conference sponsorship is to be implemented. The ninth annual *Virus Bulletin* conference, to be held in Vancouver, British Columbia, on Thursday 30 September and Friday 1 October, aims to maintain a high standard of organizational continuity and consistency from the early planning stages through to the final execution of a full conference programme, social events and the presentation of promotional merchandise.

For the first time, four major sponsors are responsible for this prestigious event. Our thanks and congratulations are due to *Network Associates*, *Sophos*, *Symantec* and *Trend Micro* – official sponsors of VB'99 ∎

Have Your Say

The face of *Virus Bulletin*, like the industry it represents, is bound to change. In our efforts to provide up-to-the-minute analyses and information, several old favourites – arguably past their prime – are set to make way for more dynamic series, opinions and images. As the new Editor I am looking to kickstart a novel trend in *Virus Bulletin*. On a daily basis, I am on the receiving end of a vast amount of opinion and attitude (most of it professional, some of it unprintable) that never sees the light of day. In my opinion, our subscribers are entitled to have a say in what is included in the pages of their industry standard magazine.

To that end, and bearing in mind the literary bent of this Editor, I am throwing open the Editorial page to you, the readers of *Virus Bulletin*. I shall print, at my discretion, any truly topical matter, however controversial or opinionated. If you have an axe to grind, email about 800 words to editorial@virusbtn.com. This is your opportunity to have your say – I look forward to hearing from you ∎

Prevalence Table – February 1999

Virus	Type	Incidents	Reports
ColdApe	Macro	763	39.9%
Class	Macro	271	14.2%
Laroux	Macro	258	13.5%
Ethan	Macro	178	9.3%
Win32/Ska	File	93	4.9%
CIH	File	54	2.8%
Cap	Macro	40	2.1%
Brenda	Macro	34	1.8%
Footer	Macro	30	1.6%
Marker	Macro	29	1.5%
Tristate	Macro	14	0.7%
Groov	Macro	11	0.6%
Nono	Macro	11	0.6%
Pri	Macro	10	0.5%
Npad	Macro	7	0.4%
Temple	Macro	6	0.3%
AntiMarc	Macro	5	0.3%
Concept	Macro	4	0.2%
AntiCMOS	Boot	3	0.2%
Argh	Macro	3	0.2%
Chack	Macro	3	0.2%
Insert	File	3	0.2%
Nottice	Macro	3	0.2%
Win32/Cabanas.A	File	3	0.2%
Win95/Hazlo	File	3	0.2%
Others [1]		72	3.8%
Total		**1911**	**100%**

[1] The Prevalence Table includes a total of 72 reports across 58 further viruses. Readers are reminded that a complete listing is posted at http://www.virusbtn.com/Prevalence/.

Seeing Triple!

The call for papers for VB'99 went out in last month's issue. Imagine our surprise when, in response, we received a *Word* document infected with O97M/Tristate.C which had been emailed from a respected European virus expert, himself a former speaker on the perils of viruses at previous *Virus Bulletin* conferences.

While not grounds for the summary dispatch of his paper, this kind of carelessness is laden with moral irony. The explanation for the 'mistake' involved a hurried response to the call for papers from an unfamiliar computer attached to a University network. Although this is highly topical stuff, this kind of approach brings to mind images of feet and injuries sustained with a shotgun ∎

IBM PC VIRUSES (UPDATE)

The following is a list of updates and amendments to the *Virus Bulletin Table of Known IBM PC Viruses* as of 15 March 1999. Each entry consists of the virus name, its aliases (if any) and the virus type. This is followed by a short description (if available) and a 24-byte hexadecimal search pattern to detect the presence of the virus with a disk utility or a dedicated scanner with a user-updatable pattern library.

Type Codes

C	Infects COM files	**M**	Infects Master Boot Sector (Track 0, Head 0, Sector 1)
D	Infects DOS Boot Sector (logical sector 0 on disk)	**N**	Not memory-resident
E	Infects EXE files	**P**	Companion virus
L	Link virus	**R**	Memory-resident after infection

Androide.969 **CER:** An encrypted, appending 969-byte virus containing the text ‘Androide 1ß by WMÆ [DAN]’. Infected files have their time-stamps set to 60 seconds.
```
Androide.969        2E8A 9E14 01BF 9E03 908B CF8D B62E 012E 301C 46B4 2ECD 21E0
```

Asahi.1040 **CR:** A prepending, 1040-byte virus containing the text ‘*.com’ and the encrypted message ‘STOP Virus (c) Red Hacker - wersja beta’. Infected files start with the word DDB8h.
```
Asahi.1040          3D10 0472 D4B0 02E8 4400 BA10 05B9 1004 B440 CD21 721E 32C0
```

David.194 **CN:** An appending, 194-byte, fast, direct infector containing the texts ‘David (variant)’, ‘DG’ and ‘????????COM’. Infected files start with the byte 4Dh (‘M’).
```
David.194           B428 B9C2 00CD 2133 C089 4521 8D3C B84D E9AB 58AB 528D 14E8
```

Dikshev.119 **CN:** A prepending, 119-byte, fast, direct infector containing the text ‘*.com’. Infected files start with the byte 91h (XCHG AX, CX).
```
Dikshev.119         C933 D2CD 213A E074 05FE C4A3 0201 B440 B977 0087 D7CD 21C3
```

Djin.132 **CER:** An overwriting, 132-byte virus with the text ‘[DjiN_DjiN] iS MaDe By THe GaBBeR’. Infected files have their date and time-stamps set to contain nothing but zeros.
```
Djin.132            B800 4233 C999 CD21 B984 00B4 40BA 0001 CD21 9933 C933 D2B8
```

Graced.1389 **CR:** An appending, 1389-byte virus containing the texts ‘COMMAND’, ‘.COM’, ‘.EXE’, ‘TLINK’, ‘TASM’, ‘BC’, ‘BCC’ and ‘Graced. Version 1.00 .’.
```
Graced.1389         B96D 0590 E890 FFE8 67FF B903 00BA FC04 E884 FFB8 0157 8B0E
```

Groupie.768 **CN:** An appending, 768-byte, direct infector containing the text ‘*.com’. The virus adds a random number of bytes during infection. The payload triggers on 11 September and infected files have their time-stamps set to 62 seconds.
```
Groupie.768         8BD0 5E81 C203 01B4 40B9 0003 CD21 B442 32C0 33C9 33D2 CD21
```

HNY.267 **CN:** A prepending, 267-byte, fast direct infector with a Russian text that translates as ‘Happy New Year 1999!’. Infected files have the archive attribute bit set.
```
HNY.267             B440 8BD5 81C2 0B01 B9FA 00CD 21B8 0042 33C9 33D2 CD21 B440
```

Innox.1333 **CER:** An appending, 1333-byte virus containing the texts ‘Innoxious Bug 2.03 ‘The Sad Flasher’ (c) 1994-96 Bugsoft-MSTU, Moscow. Dedicated to all the rock’n’roll people.’, ‘PATH=*.COM’ and ‘*.EXE’. Infected files have their time-stamps set to 62 seconds.
```
Innox.1333          33C9 8A0E 1307 81C1 3505 B440 CD21 72D4 8B0E 0B07 8B16 0907
```

Ios.1290 **CER:** An encrypted, 1290-byte appender with the texts ‘IOS initialization error.Press any key...’ and ‘COMSPEC=’. The virus recognizes previously infected files by XOR-ing two bytes at offset 0012h and offset 0013h, and comparing the result to 52h.
```
Ios.1290            FCA5 A5BF 0400 8D86 6F01 AB8C C8AB 0E1F 8DB6 7301 B9BC 04B0
```

Luci.3600 **CE:** A polymorphic, 3600-byte appender with the texts ‘OVER-X’, ‘F77\’, ‘MSFORT\’, ‘GAME\’, ‘TP\’, ‘Hard disk destroyed by mutant-virus “LUCIFER”’, ‘POLYMORPHIC ALGORITHM IS CREATED BY OVER-X’ and Russian messages. This template detects the virus in memory only.
```
Luci.3600           3D54 5475 088B D833 C9F9 CA02 0080 FC4B 7517 FBFC 1E06 2E89
```

Ocean.983 **CR:** A stealth, appending, 983-byte virus. Infected files have their time-stamps set to 62 seconds.
```
Ocean.983           3D05 4D75 04B8 FFFF CF80 FC3D 7407 80FC 4B74 05EB 06E9 8E00
```

Offi.365 **CN:** An encrypted, appending, 365-byte virus containing the texts ‘[B!Z0n //[BzZ]]’, ‘[OffsetFinder DEmO viRuS]’, ‘[Russia, St.Petersburg 1998]’ and ‘*.COM’.
```
Offi.365            073E 8AA6 4401 80F4 908D B644 018B FEB9 2C01 AC32 C4AA E2FA
```

Pkunk.1586

CN: An appending, 1586-byte virus containing the texts '[PKUNK v1.0] (c) Wet Milk', '*.*' and 'C:\*.*'. Infected files have the byte 5Bh ('[') at offset 0003h.

```
Pkunk.1586          A368 03B4 40B9 3206 BA00 01CD 21B8 0042 2BC9 2BD2 CD21 B440
```

Polish.2576

CE: An appending, 2576-byte virus containing the texts 'C:\WININI.COM', 'c:\AUTOEXEC.BAT', 'c:\abcghlkw.ftq', '.COM', '.EXE', 'COMMAND.COM', '386@lh winini.com' and the encrypted message 'Czesc jestem virus Fourlo Napisal mnie pewien madry czlowiek z Bydgoszczy lub Torunia'.

```
Polish.2576         B900 0A2E 8B1E 8209 B800 40E8 81FB 2E8B 0E88 09B1 FE2E 890E
```

PS-MPC.276

CN: A 276-byte, fast, direct infecting appender with the texts 'SST98', 'Annihilator' and '*.cOm'.

```
PS-MPC.276          B43F FEC4 8D96 0301 B914 01CD 21B9 0042 B800 0091 BA00 00CD
```

Quiz.494

CER: A prepending (COM) and appending (EXE) 494-byte virus containing the texts 'QUIZ' and '*.*'. Infected COM files start with the word 1E56h.

```
Quiz.494            BAEE 01B4 3FB5 FDCD 218B FA81 3D56 1E74 2780 3D4D 0F84 C100
```

Replicator.454

CR: An appending, 454-byte virus containing the text 'V1.00' and a Russian message. Infected files end with the byte 56h ('V').

```
Replicator.454      3DCD AB74 0D3D 004B 7503 E842 002E FF2E C101 B814 FFCF 0D0A
```

Second.697

CN: An appending, 697-byte, direct infector containing the text '(c)The Second*.com'. Infected files have the third last byte set to E9h.

```
Second.697          B440 B9B7 02CD 21B9 FFFF 5B2B 4F0A 81E9 BB02 894F 0A8B D383
```

Sirius.361B

CN: An encrypted, appending, 361-byte virus containing the texts '*.CoM' and '<ix>'. Infected files have their time-stamps set to six seconds.

```
Sirius.361B         8D76 1A90 E802 00EB 108A 9663 01B9 4901 8BFE AC32 C2AA E2FA
```

Skate.215

CN: An appending, 215-byte, fast, direct infector containing the texts '*.COM' and 'Skate'. Infected files have their time-stamps set to 17 May 1992.

```
Skate.215           B440 B9D7 008B D681 EAC3 00CD 2190 B43E CD21 8BD6 83C2 3190
```

Sperm.791

ER: An encrypted, appending, 791-byte virus containing the texts '<I'm merry SPERM v2.5>' and '*.exe'. Infected files are three bytes shorter than the size information recorded in the MZ header.

```
Sperm.791           D7BB E660 B9B0 F550 5351 8926 0400 CD01
```

Starr.688

CN: A 688-byte, direct infecting overwriter with the texts 'Your system already infected by Starr virus! Your system already infected by Starr virus! Your system already infected by Starr virus! Your system already infected by Starr virus! Your system already infected by Starr virus! SSSS TTTTTT AAA RRRRRR RRRRRR SS TT AA AA RR RRR RR RRR SSS TT AA AA RRRRRR RRRRRR SS TT AAAAAAA RR RR RR RR SSSS TT AA AA RR RR RR RR Your system already infected by Starr virus! Your system already infected by Starr virus! Your system already infected by Starr virus! Your system already infected by Starr virus! Your system already infected by Starr virus!' and '*.com'. The 'SSSS TTTTTT…' string forms a multi-line banner.

```
Starr.688           BA9E 00CD 2193 B440 B9B0 02BA 0001 CD21 B43E CD21 CD20 B409
```

Taek.2119

CER: An encrypted, appending, 2119-byte virus containing the text 'Blue Scorpion | Copyright (C) 1995-1996 Taek Software. | Ver Blue.2119'. Infected files have their time-stamps set to 54 seconds. The following template detects the virus in memory only.

```
Taek.2119           B440 B947 0806 1F33 D2E8 2DFC 7244 0E1F 803E 3F07 0074 10B4
```

Trivial.588

CN: An overwriting, 588-byte, fast, direct infector containing the texts '!<<-NoFx->>!', ' [New BaBy Born and U will LoVe HiM So Much!] =[The FiRst OveRWriting BaBy]= ', 'Best PunkRock band all time' and '*.com'. Infected files have the byte 61h ('a') at offset 0003h.

```
Trivial.588         B440 B94C 02BA 0001 CD21 B43E CD21 B44F EBBA B409 BA82 01CD
```

Tunneler.811

CN: An encrypted, 811-byte appender with the texts '.exe', '.com', 'scan', '000', '-d', 'alik', 'all', 'anti', 'arj', 'asta', 'avast', 'avg', 'avp', 'chk', 'clean', 'comma', 'cpav', 'disc', 'disk', 'dizz', 'dummy', 'f-', 'find', 'fv', 'goat', 'guard', 'hell', 'ibm', 'kill', 'mem', 'ms', 'msav', 'nav', 'ndd', 'nod', 'pk', 'rar', 'rex', 'safe', 'save', 'shiel', 'stop', 'tb', 'test', 'tbav', 'tc', 'vir', 'test', '[wRITTEN 1994-5 bY tHE tUNNELER/uSA/cA - vERSION 1.00]', 'gREEtZ tO lITTLE jOE!' and '*.C*'. Infected files have their time-stamps set to 62 seconds.

```
Tunneler.811        4901 56FC B97E 018B FE8B 9645 04AD 33C2 D1C2 03D1 ABE2 F6C3
```

Vampiro.1495

CER: An encrypted, stealth, appending, 1495-byte virus containing the texts 'CHKLIST.MS', 'ANTI-VIR.DAT', 'TBF-SC', 'NO NO NO!! Que hace todavia despierto?!?!', 'Drako & Zarathustra le OBLIGAN a dormir.', 'Que sea la última vez, ok?!?! En caso contrario, morira.. JAJAJAJAJAJA' and 'Vampiro 2.0 Versión de Batalla'.

```
Vampiro.1495        B400 CD1A 8B6E FA81 ED08 01BA AD05 8BCA 2E8A 961E 01BF 2901
```

Wormsign.1575

CN: An appending, triple-encrypted, 1575-byte virus containing the texts '*** THE SANDWORM ***', 'Wormsign !' and '*.com'. Infected files have the byte 2Ah ('*') at offset 0003h.

```
Wormsign.1575       2F03 03FE 8AA4 1203 B9FD 058A D480 E20F F810 2502 E247 E2F9
```

VIRUS ANALYSIS 1

Happy Gets Lucky?

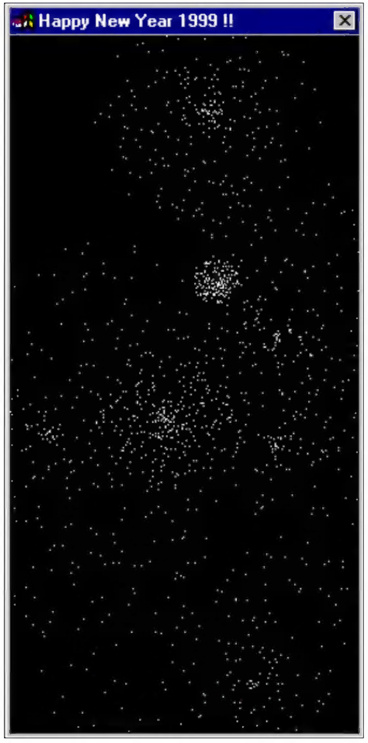
Péter Ször
Data Fellows

Virus writers want their creations to spread as far and as fast as possible. Several of them have attempted to use the Internet to achieve this goal. A few years ago they simply tried to spam infected files to newsgroups. While very few were really successful in this, some of them became infamous because of it, especially a French virus writer called Spanska. His early DOS creations even became in-the-wild viruses.

Spanska's strategy was much better than average when it came to spreading his viruses. He posted infected hack programs to certain newsgroups which could be used to register commercial products. Thousands of people look for such hacks on the Internet every day, infecting their computers with the attached virus. Unfortunately, this all happened in the past – the present is automation.

More and more viruses are written with built-in network features. Win32/Parvo (see *VB*, January 1999, p.7) was the first virus which could spam hoax messages with infected attachments by using socket communication. Parvo was not very successful due to certain limitations.

We knew Parvo was the first, but would not be the last, of its kind. During January 1999, Spanska's new creation, Win32/Ska, became well-known by thousands of Internet users the world over.



It is not particularly easy to classify Ska. While most virus researchers agree that it is a worm, others, including myself, have some doubts about this. The fact is that Ska cannot be classified as a real virus and it is not a traditional worm either. The general consensus is that its working mechanism shows more similarities with worms.

Ska gets inside a computer via an email or newsgroup attachment, affecting those machines that run the attachment. If an unauthorized attachment called HAPPY99.EXE is run, Ska puts up an attractive fireworks display, which could easily be mistaken for a good-looking accessory to the message. However, when the fireworks burst on-screen, Ska has already modified the WSOCK32.DLL file (if it was available for access) in order to monitor all postings that are made from the machine. All Internet access goes through APIs placed in the WSOCK32.DLL.

Afterwards, Ska spams the newsgroup or email recipient with copies of itself any time the user tries to send a message across cyberspace. Two messages are posted from the machine each time – the original mail and a copy of it with a UU-encoded attachment called HAPPY99.EXE which goes to the same location. This attachment is exactly 10,000 bytes long when decoded.

Ska shares many similarities with chain letters. Chain letters are classified as worms. When someone receives a chain letter with attachments (most often a script), the executed attachment can look for other email addresses and post itself to those places without modifying anything on the local machine. Ska works that way, with one important difference. It modifies a local DLL in order to get control later. It does this by modifying the WSOCK32.DLL file (it does not patch the DLL in memory). Therefore, rebooting the attacked PC will not remove Ska from the machine.

Executing HAPPY99.EXE

In the first stages, a user receives two messages similar to the ones we have seen on samples@datafellows.com:

```
Date: Fri, 26 Feb 1999 09:11:40 +0100 (CET)
From: "XYZ" <xyz@xyz.cz>
Subject: VIRUS
X-Spanska: Yes
```

Note the 'X-Spanska: Yes' in the header. This is added by Win32/Ska. Everything beginning with X- is simply skipped by the mail servers. The body of the mail is a UU-encoded attachment.

The original message reads:

```
From: "XYZ" <xyz@xyz.cz>
To: <samples@datafellows.com>
Subject: VIRUS
Date: Fri, 26 Feb 1999 09:13:51 +0100
X-MSMail-Priority: Normal
X-MimeOLE: Produced By Microsoft MimeOLE
V4.72.3110.3
HAPPY NEW YEAR 1999.
I delete the attachment but it seems to still
block my computer.
What can I do. Thanks for your reply
```

When the user executes HAPPY99.EXE, Ska is activated.

Initially, it allocates memory for its own use. If that should fail, Ska will terminate immediately. After that, it checks if the system is Win32-compatible and terminates if it is not. Then it decrypts a short encrypted area of itself which contains the following text:

```
Is it a virus, a worm, a trojan? MOUT-MOUT
Hybrid (c) Spanska 1999.
```

After this, it gets the exact location of HAPPY99.EXE and copies it into the *Windows* system directory as SKA.EXE, unpacking an area into the previously allocated buffer. The unpacked code is used to create SKA.DLL (8192 bytes) in the *Windows* system directory. This DLL has two exported APIs – 'mail' and 'news', respectively.

Next, it makes a copy of WSOCK32.DLL, renaming it WSOCK32.SKA in the *Windows* system directory and tries to open WSOCK32.DLL for write. Ideally this is possible if the DLL file is not set to read-only and it is not in memory. Ska does not handle the read-only flag – setting the file to read-only can save WSOCK32.DLL from the patching. If the DLL is in memory, the file cannot be opened, and Ska tries to modify the registry field ...CurrentVersion\RunOnce to execute SKA.EXE from there during next boot (when WSOCK32.DLL will not be loaded). If WSOCK32.DLL is not in memory yet, then Ska tries to patch it.

It checks if WSOCK32.DLL is already patched by comparing the checksum field of the DOS EXE header to 7Ah. (If the checksum is not 7Ah it sets it to this value which is its self-recognition mark). Then it searches over the section table and makes the necessary modification. It sets the .text section to writeable and makes the virtual size of it a little bigger. This is in order to fit in there, in the section slack, with its short hook routines at the end of the .text section. Then it searches for two API names in the WSOCK32.DLL export sections. These are called 'connect' and 'send'. It patches the export address entries of these APIs to point to new entries at the end of the .text section.

Next, it gets the address of a few APIs and saves pointers to the entry point of those functions which will be used later. Finally, it patches a very short routine at the end of the .text section in WSOCK32.DLL. Then the patch is ready. After this, Ska calls the message box animation routine, but only if HAPPY99.EXE was executed. If the SKA.EXE copy is executed from boot, the animation is not displayed.

Posting a Message

When the patched WSOCK32.DLL is loaded in memory Ska intercepts the 'connect' and 'send' APIs. When the user makes a connection to anywhere, Ska checks the ports utilized. If it is mail or news port access Ska loads the SKA.DLL in memory. When a DLL is loaded, its initialization entry point is called. In this initialization routine another encrypted area is decrypted in the SKA.DLL which is related to the uu-encoding and a large enough buffer is allocated for future use. At that point, the original 'connect' entry point is called from WSOCK32.DLL.

When the intercepted 'send' is called, Ska re-checks if the sending is news- or mail-related by getting the addresses of 'news' or 'mail' APIs from SKA.DLL and calling them first. It copies part of the original email header to a new buffer, paying attention to 'NEWSGROUPS:', 'MAIL FROM:' , 'TO:', 'CC' and 'BCC' and copying them for its own use. Finally, it adds the 'X-Spanska: Yes' string to the existing mail header.

It opens SKA.EXE and converts it into uu-encoded form in between the 'begin 644 Happy99.exe' and 'end' lines which marks the uu-encoded attachment. It posts this mail to the specified location. After this the original 'send' request is called which posts the original message without the HAPPY99.EXE attachment.

Ska logs the addresses where it could post itself. A text file called LISTE.SKA is created for that use in the *Windows* system directory. This text file will list a limited number of locations where Ska could post itself successfully.

Conclusion

We are bound to see more and more viruses and worms capable of posting themselves all over the place. That means a new challenge for anti-virus researchers.

Very often we have to spend hours, days or a full week with some creations until we can provide a solution. Virus writers create something new again in a couple of months. The typical user wants a 24-hour solution. Always difficult, this is becoming even more challenging this year, especially when users still insist on executing uncertified attachments received from their 'friends'.

Win32/Ska

Aliases:	Happy99, SKA.A, I-Worm.Happy, MOUT-MOUT.
Type:	Win32 worm.

Self-recognition in WSOCK32.DLL:

Sets the checksum field of the DOS EXE header to 7Ah.

Hex Pattern in WSOCK32.DLL and HAPPY99.EXE:

```
3319 7508 8B44 2428 AA47 EB0A
3C77 751B 478B 4424 28AA E808
0000 0053 6B61 2E64
```

Intercepts:	Hooks Send and Connect APIs of WSOCK32.DLL.
Payload:	When executed as HAPPY99.EXE, it displays fireworks animation.
Removal:	Delete SKA.EXE and SKA.DLL and replace WSOCK32.DLL with WSOCK32.SKA in the *Windows* systems directory.

VIRUS ANALYSIS 2

Virtual Tomatoes

Shane Coursen
WildList Organization

Do you give presentations? No doubt you will always take great care to rehearse, to make sure that your body language is just so, as well as to ensure that your pauses are long enough to produce the intended dramatic impact, and that you do not go over your allotted time. To be sure, it is a lot to remember when standing in front of your peers. According to Murphy, a public presentation is an event where many things can go very wrong, very quickly.

Now there is a new gremlin that just may make presenting a bit worse. It is a computer virus that infects *PowerPoint* presentation files – one whose payload activates *during* the slide show. As if presenting were not difficult enough already, we now face virtual tomatoes thrown at us by our own presentation design tools!

Residency and PowerPoint Presentations

When a virus goes resident in memory, it is usually said to have hooked into something – a software interrupt, for example. With classic file and boot viruses, the hook is at a very low level (usually the interrupt table), thus giving the virus a tremendous amount of control. Were there no operating system, the virus would not be able to hook into, go active, or go resident in memory. *Office* provides the 'operating system' for macro viruses – they 'hook' into *Office* functions and so gain a type of memory residency.

Just like with file and boot viruses, an *Office* macro virus' residency is limited to the existence of its *Office* parent application – *Word*, *Excel*, *PowerPoint*, etc. The residency can be further limited to specific times when the application is open. For example, certain mouse actions are detectable and/or programmable only during a slide show.

Another, more specific example, is when you advance to the next slide during a slide show. The typical action would be to press the mouse button. The viruses covered in this series gain residency in just such a manner – by hooking into the mouse click action. For example, if PP97M/Shaper is 'resident' during a slide show, instead of advancing to the next slide when the mouse button is clicked, a macro named 'actionhook' is called.

PP97M/Vic.A

PP97M/Vic.A has no payload. It is very short and to the point and unlike the other two viruses covered in this series it does not add a Module. Instead, Vic inserts (or prepends) its code via the CodeModule InsertLines method into an existing user form. If the presentation does not contain a user form, it is not a viable infection target. Vic does not create a new user form. Before Vic can execute, an infected user form must be running. There are many methods for initiating a user form, too many to discuss here, so we will assume that a Vic-infected user form is already running.

Opening a user form is one thing and closing it is another. Closing a user form is performed by clicking on the X (close current window) in the upper right hand corner of the form. When this X is clicked, a pre-defined function named UserForm_Terminate() is called. When Vic inserts its code into a user form, it rewrites UserForm_Terminate(). By doing so, calling UserForm_Terminate() now causes Vic's code to execute prior to the user form actually closing down. It is when a Vic-infected user form is closed that Vic's replication code is activated.

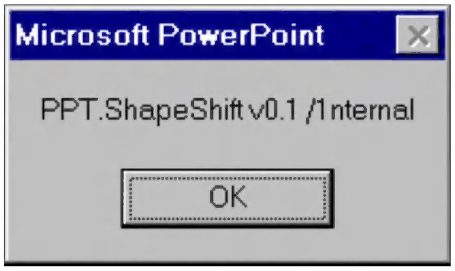
It should be noted that if a redefined sub function named UserForm_Terminate() already exists, it will not be overwritten when Vic infects. Rather, it will simply be 'moved down' when the virus inserts its code. Nevertheless, the pre-existing UserForm_Terminate() code will not be called upon as long as a function defined with the same name is 'higher' up in the code path.

When activated, Vic immediately searches out *.PPT in C:\My Documents and all subfolders of C:\My Documents. Vic checks each .PPT file found for the existence of a type 3 object – a user form. If a user form exists, Vic tests the form (also via CodeModule – the Lines method) to see if the first line of the associated macro equates to "'<!—1nternal—>" – Vic's self-identification routine. If so, the file is considered infected, so Vic continues to search for the next PPT file in the search list.

To make a visual determination that a presentation is infected with Vic, start by opening the Visual Basic Editor. With VBE open, look in the Project Explorer List window. Expand the suspected presentation's project (typically VBAProject) folder. If you see nothing new after expanding the project folder, you can stop. If you see other items, keep going. If one of the items that you see is a Forms folder, expand it. If there are any user forms defined underneath, start viewing the code of all associated macros. Specifically, look for Vic's self-identification string at the first line of each macro.

PP97M/Shaper.A (aka ShapeShift)

This virus does have a payload. During the slide show, if you click on a hooked AutoShape, there is a one in ten chance that an AutoShape on the last slide of the presentation will be deleted (although as explained later, it is not so straightforward). There is also a one in ten chance the following message box will be displayed.


Microsoft PowerPoint

PPT.ShapeShift v0.1 /Internal

When going to the next slide during a slide show, the most typical action would be to press the mouse button. Shaper gains residency by hooking into the mouse click action. If Shaper is 'resident' during a slide show, instead of advancing to the next slide when the mouse button is clicked, a macro called 'actionhook' is called.

The actionhook macro contains all of Shaper's viral code. If, upon entry into actionhook, the user is exiting the last slide during a slide show, one or more events may occur. Immediately, Shaper checks all open presentations in turn to see if they are infected. To determine prior infection, Shaper checks each presentation for the existence of a module named ShapeShift. If a module by that name exists, the presentation is already infected.

If the ShapeShift module does not exist, it soon will. Like ShapeMaster (see below), Shaper utilizes CodeModule. Initially, Shaper creates a module called ShapeShift, then calls upon the InsertLines method to copy the viral code into the newly created module.

Then the hooking begins! Shaper first checks all of the AutoShapes on the last slide of the target presentation for one named hookme. If an AutoShape by that name is not found, two things will happen.

First, an AutoShape with the name 'hookme' is created. The chosen shape is randomly selected from the available list of AutoShapes (and based on a seed value of 140). The height and width characteristics for the new AutoShape are the same as defined in the slide's PageSetup area. Then Shaper attempts to hide the AutoShape by making it invisible (it has no line colour and no fill colour) and by sending it to the background.

Again, just like ShapeMaster, Shaper hooks the shape by setting its Action Settings to run actionhook. In this phase of infection, the most obvious difference between Shaper and ShapeMaster is that Shaper places the AutoShape directly on to the slide (whereas ShapeMaster places the AutoShape on the Slide Master).

Next, more AutoShape hooking occurs. After creating the new AutoShape, the virus attempts a systematic hook of all the remaining AutoShapes on the slide. All non-zero (unused) AutoShapes are hooked. In layman's terms, this means that Shaper will hook every AutoShape on the slide that does not already have an assigned mouse click action.

From this point forward, when the presentation is run, if the presenter clicks on one of the hooked AutoShapes, the virus will run (possibly infecting more *PowerPoint* presentations and hooking more AutoShapes). Once complete with all the hooking work on the last slide, the virus yet again makes two more passes, each pass hooking more AutoShapes on different slides.

On each pass, a random slide is selected. The randomness is based on the total number of slides that exist in the presentation. Then, from the randomly selected slide, an AutoShape from the slide (if any exist) is chosen, again at random. The randomness of the AutoShape is based upon the number of AutoShapes that exist on the selected slide.

Bear in mind that even text boxes are considered AutoShapes. This means that nearly everything on a slide can be hooked – it does not have to be a graphic shape or clip art file! If you have already assigned Action Settings to a text box (or other AutoShape) to execute a macro, those settings will not be lost when Shaper infects. Shaper can avoid overwriting any existing AutoShape Action Settings by checking each AutoShape's ppActionMacro value.

It would make sense that if one were to delete an AutoShape deliberately, one would want the 'damage' to stay permanent. In the case of Shaper, the payload (more specifically, the point where the payload deletes the 'hookme' AutoShape it created earlier) seems to be more a method of just changing its appearance now and again.

There is a one in ten chance the call to delete the last slide's hookme AutoShape will go out. When it does, the shape appears to be deleted. Damaging, right? No, not really. Remember, *it* created the AutoShape named hookme to begin with. However, immediately after the call to delete hookme is made, the program creates a new hookme (again on the last slide using PageSetup, as described earlier).

The effect of this is, for example, that the hookme may change in shape from a rectangle to an explosion. After the shape change, there is a one in ten chance the ShapeShift message box will be displayed.

To make a visual determination that a presentation is infected with Shaper, in Slide View, right click on any AutoShape in the slide. It does not have to be the classic GIF, JPG, or ClipArt graphic – in *PowerPoint*, the text boxes where you record your slide titles and bullet points are considered AutoShapes. Next, select 'Action Settings'. Make sure the Mouse Click tab is on top and look for the 'Run Macro' radio button.

If the name being pointed to is 'actionhook', this presentation may be infected with the Shaper virus. Clicking on the arrow to the right of it displays all the sub functions defined within the ShapeMaster module –'RandomWackSlide', 'SlideIn', 'WackPresentation', and 'WackShape'.

PP97M/ShapeMaster.A

With this particular virus, there is a one in ten chance the mouse click action during a slide show will be reversed. Instead of advancing to the next slide, a mouse click may step to the previous slide. Along with the unexpected slide display order, there is a one in five chance of a message being displayed. Normally, to advance to the next slide during a slide show, one would simply click the mouse button. As described in the opening comments about residency, ShapeMaster gains a type of residency by hooking into the mouse click action during a slide show.

If ShapeMaster is resident when the mouse button is clicked during a slide show, instead of advancing to the next slide, a macro named 'actionhook' is called. This macro contains ShapeMaster's viral code. When actionhook is activated, there is a one in seven chance the virus will search out and infect *PowerPoint's* Blank Presentation template file (Blank Presentation.pot – typically located in the subfolder \My Documents\Microsoft Office\Templates). Blank Presentation.pot is the only file that an infected ShapeMaster presentation will infect. This particular virus does not infect directly from one infected *PowerPoint* presentation to another.

Once situated in the Blank Presentation template file, ShapeMaster increases its chances of becoming part of all subsequently created *PowerPoint* presentations. The Blank Presentation template file acts in a fashion similar to NORMAL.DOT in *Word*. In *PowerPoint*, when a new presentation is created – *if* it is based upon the Blank Presentation template – it is based upon the template file Blank Presentation.pot.

Assuming the one in seven chance evaluates to true, ShapeMaster builds a path to Blank Presentation.pot. Once completed, it tries to open the file. If Blank Presentation.pot does not exist, the virus will continue unawares, attempting to write its code to the nonexistent file. Nevertheless, even with the absence of the blank template file, ShapeMaster continues on without any noticeable errors; actionhook will terminate normally.

Assuming Blank Presentation.pot *does* exist, ShapeMaster looks at each of the module names already held within. If a module by the name of ShapeMaster exists, the file is assumed to be infected already, so the virus replication routine is terminated.

If a module named ShapeMaster does not exist it means Blank Presentation.pot is not infected. ShapeMaster proceeds to do just that by first creating (adding) a new module, naming it ShapeMaster, and finally attaching actionhook's code to the module. If other modules exist, the ShapeMaster module finds its place alphabetically among the list of module names (listed in VBE).

ShapeMaster then adds a shape (AutoShape #1 – a rectangle) to the Slide Master of Blank Presentation.pot. Scaling it to the same width and height as the presentation's page setup values, the newly added AutoShape is exactly the same size as the viewable slide. ShapeMaster also sets the new AutoShape's colour and fill values to zero, making it invisible to the casual viewer.

ShapeMaster also assigns specific Action Settings to the Slide Master. These apply to what happens when an AutoShape is 'clicked on' or 'moused over' during a slide show. In all cases, the virus assigns the AutoShape to run the actionhook macro.

Albeit benign, ShapeMaster *does* have a payload, and it is during the presentation that it might activate. There is a one in ten chance that during a presentation, when the mouse is clicked, you will step back to the previous slide. Nestled within the reversed slide payload routine is yet another mini-payload – a message box. There is now a one in ten chance a message box similar to Shaper's will be displayed.

Placing the AutoShape on the master slide using PageSetup values has a few advantages. If, for example, the virus author placed something more conspicuous – say the explosion shape – on the master slide, and gave it colour and fill, the likelihood of it being noticed would increase. Making the conspicuous shape invisible is the obvious answer, however, any shape other than the rectangle would decrease the 'infectious surface area' with which to work. In the real world, a noticed virus is usually a dead virus. Instead of the theoretical explosion shape, AutoShape #1 (a rectangle) is selected, scaled, and made invisible.

To detect this virus visually, start by editing the Slide Master and select all objects. You will see that each item has several handles that you can grab to resize, etc. Look for the handles corresponding to an item stretched to the very limits of the physical slide. If you see such handles, first press Esc to deselect all shapes. Then place your mouse pointer near the outer edge of the slide master, and click once. You should see the handles of the stretched shape (and only that one shape) reappear.

Now right click and select Action Settings. In the Action Settings box, you will see the 'Run macro' radio button. The radio button may not appear to be activated, but if the Run macro being pointed to is 'actionhook', the presentation may be infected, as will most likely be *PowerPoint's* Blank Presentation.pot template file.

Summary

CodeModule is a very powerful add-in offering Visual Basic programmers many different powerful capabilities. Naturally, CodeModule offers virus writers those same capabilities. It is not a great surprise that all three utilize the CodeModule.InsertLines method.

Needless to say, the three viruses covered in this analysis were initial attempts at a previously unexplored idea. Most experts agree they will never spread far beyond the lab. This may cause us to believe they do not present any danger. In the immediate sense that may be true, but in another it is quite the opposite. In fact, these might be the most dangerous *PowerPoint* viruses ever to emerge – they represent the seeds from which new and more dangerous ideas may develop.

FEATURE 1

The Virus Analyst Headache

Eugene Kaspersky
Kaspersky Lab

The careful analysis of new viruses appearing every day is the main indication of an anti-virus scanner's quality. Automatic or semi-automatic analysis, adding the detection and disinfection records to the anti-virus databases, may cause low detection rates and/or missed infections.

Then the scanner reports that all the files on the system are disinfected, but on next reboot the virus appears again and reinfects everything. Users are not happy with that, especially if the virus has its 'doomsday', erasing all their data so that the only way to get rid of it is by reformatting destroyed hard drives.

Sometimes, this does happen. Imagine a new variant of DOS virus which infects COM and EXE files, but which also infects SYS drivers. A careless analyst may miss this new feature, and SYS files will stay undetected. Then the scenario described above ensues. This is why it is necessary to pay attention to all virus branches and routines; each of them may have unexpected outbreaks. If an analyst misses this kind of thing, the virus may find its way into the wild.

Careful Analysis – Is it a Big Deal?

Ten years ago Jerusalem and Cascade were the 'scary monsters' which anti-virus experts spent days disassembling and trying to understand. These were the green years of the anti-virus industry – dreamland now – when new viruses appeared once a week. Several hundred new viruses and variants each month – that is the reality on today's virus conveyor belt.

That means that the average virus analyst has to process about ten or more viruses per day, if the lab employs around five virus experts. Do not forget that the same virus experts usually elaborate and support scanning and disinfection engines, and they also want weekends off and holidays.

It is pointless to say 'Hire more virus analysts' – it is very hard to find an experienced virus analyst (without a virus-writing past). Inaccurate processing by an inexperienced analyst will cause false positives and negatives. That will necessitate the hiring of more tech-support people, and even the occasional high-class expert to clear bugs in your company's anti-virus databases.

When a virus analyst's career starts and they analyse their first virus – that is a challenge. The first dozen is an interest. The first hundred is a hobby. The first thousand becomes a routine, a conveyor belt of viruses moving quickly from the Incoming to the Sorted area. Open a new one, disassemble and glance inside it, see that it is a variant of a known virus – this is nothing new. Until the virus conveyor belt stops. It stops not because the Incoming area is empty and there is nothing to do (I dream about that!), but because something complex and new has appeared.

Ancient History

Virus analysts who started out in 1990 may remember two DOS viruses which turned an ordinary day into a nightmare. They were Whale (a variant of Fish#6) and Pogue (based on the MtE – the first strong polymorphic engine).

The 9 KB Whale virus looked like a very complex addition to the DOS kernel – it hooked about 20 DOS functions, and corresponding virus subroutines ran the infection and stealth virus mechanisms. That was enough of a challenge in those days: locating all the hooks and infection routines, getting past anti-debugging tricks. Several days were lost just getting used to working with this kind of virus.

Pogue presented anti-virus scanners with a new generation of polymorphic virus, and as far as I remember, it stayed undetected by any of them for several months. Virus experts had to choose the way to analyse it: either by replicating several thousand samples and using statistical methods, or by analysing the very intricate polymorphic engine's subroutines.

A Whole Lotta Viruses

Imagine you receive a several megabyte archive full of new viruses – about fifteen thousand of them. Fridrik Skulason (*FRISK Software*) was the first to receive such a nightmare package, passing it onto other experts saying 'Let me ruin your day'. It was no big deal to write a generic detection and disinfection routine for all these samples, but it still needed to be tested – they all had to be replicated, and run against detection and disinfection tests. That meant that it was necessary to open and copy a sample, copy 'sacrificial goat' files, run the virus, infect the files, move them to a '\Replicated' directory. The computer had to be rebooted after that to be sure that there were no virus traces left in the system. Try repeating that fifteen thousand times.

That was a huge task. In my case, the virus-replication computer (a *Pentium-130*) was working with no long interrupts for about a week, and several times the hard drive ran out of disk space, overflowing with infected samples.

New Platforms and Formats

It is not easy constantly switching your focus to new types of virus. Boot and DOS parasitic viruses evolved into *Windows* viruses, then macro infectors, and then selfreplicating Java applications, VisualBasic scripts, HTML pages, and so on. Nowadays, viruses occupy all niches in the computer's 'biology'.

At first, viruses infect new popular and modern (at the moment they appear) operating systems (*Windows*, *OS/2*, *Linux*). It is necessary to have the right tools to disassemble them, and to be informed about internal executable file formats. That is relatively easy.

Unfortunately, they often have additional features, and to locate virus code, in some cases, it is not sufficient to find out the address of the program's entry routine. The virus code can be linked with other parts of segmented new executables – to file Exports. Win/RedTeam (see *VB*, May 1998, p.6) affects exports in Win16 NE files, Win32/SKA (see p. 6 of this issue) exports in Win32 PE files. It is also unfortunate that *Windows* viruses with quite simple internal file structures run under quite complex environments.

Win32 kernel's internal formats and features are not described in any documentation, but we need them to add detection and disinfection for memory-resident *Windows* viruses, and these formats are different again from good old, well-known DOS MCBs (Memory Control Blocks). Needless to say, it is a good idea to have in mind formats of protect-mode Global, Local and Interrupt Tables – often they help to understand what the virus does.

The number of viruses (including those discovered in the wild) depends on the popularity of the operating system. Fortunately, now we have only one – Win32. Imagine that any other (say, *Linux*) will be also very popular, and incoming *Windows* stuff will be doubled by *Linux* viruses.

The Macro Problem

We have to put up a big flag here with *Microsoft Office* written on it. The internal binary formats of *Office* documents, sheets, presentations and other components are much more complex than *Windows* file formats and disk space allocation tables. To detect and disinfect macro viruses the anti-virus scanners have to support these formats, so anti-virus experts have to be familiar with all of them. These formats are undocumented, and anti-virus labs have to start their own investigations to build this knowledge-base. That is why 'true' detection and disinfection methods were only embedded into anti-virus scanners six months after discovering the first Concept macro virus.

This chapter is not finished. There is more room for macro viruses now and in the future (for instance, VBA is licensed for use in *CorelDraw*).

High-Level-Language Viruses

Up-to-date disassemblers are familiar with most DOS HLL (High Level Language) executable files written in C/C++ or Pascal, and work with such DOS viruses is no more onerous than with average viruses written in assembler.

The disassembler detects the compiler which was used to compile the virus, loads necessary libraries database, locates main program's routine and comments all calls to runtime library.

It is not the same for all Win32 compilers. There are several that generate 'black boxes' for the analyst. Delphi and the latest VisualBasic compilers produce easily comprehensible executable files. Imagine an average Delphi program (just 500KB). There is no really good tool to disassemble it, disassemblers just output several megabytes of pure commented code. Sometimes it is quite difficult to separate viruses written in Delphi from non-viral programs. The virus analyst must run it on the test computer, watch its behaviour, and log results. It is not hard to see why this is not the best way.

Terrible Tricks

This last section is dedicated to the special tricks that virus writers add to their creations. The SSR, Zhengxi and Nutcracker viruses, to name a few, are fat, complex, often stealthy and extremely difficult-to-analyse programs. They use many anti-debugging and anti-disassembling tricks like on-the-fly en/decryption, hidden branches – everything virus writers can imagine to make virus analysts frustrated.

The Lexotan, TMC and some other viruses use self-mutating algorithms. That means that the virus is not encrypted, but its whole 'working' code is mixed with junk instructions. The virus changes the sequence of routines and branches, mutating data offsets in its assembler instructions, constants and so on.

The Latest Thing

A new *Windows* virus I received recently turns these tricks against the *Windows* platform. This polymorphic, Win32, memory-resident virus, named Harrier after the text in its body, appeared to have about 100 KB (yes, one hundred kilobytes!) of assembler code.

It stays in memory as part of an infected program, hooks about 30 (that is correct, thirty!) *Windows* functions, manipulates PE files sections and Import tables, and so on. Even after several layers (from 9 through 17) of polymorphic decryption loops have decrypted the virus code step by step, the virus routines do not appear in 'easy-to-analyse' form. All virus instructions (about four thousand lines of assembler code) are randomly mixed in the virus code and linked by JMP opcodes.

Needless to say, it is impossible to analyse the virus in this form, and it is necessary to assemble its disassembler to ordinary readable state. I spent about 10 hours 'compressing' the virus disassembler and used specially developed helpers, but anyway that was a crazy task. This kind of virus is not the thing virus experts dream about, but it happens, and when it does, no amount of pain killers will ease your headache!

FEATURE 2

Letter from South Korea

Charles Ahn, MD, PhD
Dr Ahn's Laboratories

South Korea has over 50 million inhabitants and 4,332 years of history. The 1988 Olympic Games were held in its capital, Seoul, and it is to co-host the Football World Cup in 2002 with Japan. In the world of IT, Asia's market has huge development potential and Korea itself is the second biggest market to Japan.

It is impossible to discuss the history of the Korean anti-virus software industry without mentioning my role in its conception. My personal history is unique and well-known to Koreans. I got an MD and a PhD from medical school, but decided to change my job from a medical doctor to a 'computer doctor'.

In 1988, I wrote the anti-virus software program *Vaccine* in order to eliminate the Brain virus – the first ever found in Korea. *Vaccine* has been continuously upgraded to *V2*, *V2plus* and *V3*, all available to the public free of charge. Until the middle of the 1990s, I had worked as a medical doctor, but also as the only anti-virus researcher in Korea. Consequently, in Korea, even computer illiterates recognize the name Charles Ahn and *V3* when they hear about computer viruses.

In March 1995, I quit my medical activities and established *Dr Ahn's Laboratories*, of which I am President and CEO, to do full-scale business. In the beginning, *Dr Ahn's Laboratories* released *V3Pro 95*, a commercial anti-virus software program for use with *Windows 95*.

Four short years on, *Dr Ahn's Laboratories* is marketing its own 'total anti-virus solution' for *Windows 3.x/9x/NT* desktop, *NetWare* server, *Windows NT* server, Unix server, email server, *Lotus Notes* server, *MS Exchange* server, HP OpenMail server, Internet gateway… We have also developed a network management tool for anti-virus software, aimed at reducing maintenance costs.

As one of the first and only Korean anti-virus companies, *Dr Ahn's Laboratories* has been on top in terms of market share since its establishment. Our market share in 1998 was 75%. Based on its accomplishments in the Korean market, *Dr Ahn's Laboratories* is now preparing to export to other Asian countries such as China and Japan.

Early Days

It was during the 1988 Seoul Olympic Games that the first computer virus made its attack on Korea. Numerous users suffered heavy losses at the hands of an unidentified virus, later known as Brain.

No other viruses came along for a year after that, so people tended to forget the damage caused. However, with the sudden appearance of the LBC (NjhtoLbc.Korea) virus in August 1989, we soon got used to the idea of virus chaos again. Hard disks infected by this virus were neither bootable nor recognizable. Most users had to reformat them, and thus, their existing data was lost. This was one of the most harmful computer viruses to hit Korean users until the beginning of the 1990s.

From then on, familiar viruses such as Jerusalem, Sunday, Stoned, Pingpong and Cascade started to flow in, and as time went by, Korean-made viruses started to appear. Honey, the first Korean virus, was found in the first half of 1989, but it did little damage to the public because it only infected floppy disks.

Opening the Source

Dark Avenger.1800 and Invader (AntiCAD.4096) were the viruses that did the most damage here in 1990. With the appearance of these two viruses, computer-related books started to be published, among them publications covering virus codes and sources verbatim.

Through this medium, the sources of boot viruses such as Brain, LBC and Stoned, and file viruses like Jerusalem and Vienna were made available to the public. This, obviously, went a long way to encouraging virus production, providing good references for Korean virus writers. It was exactly this kind of background that produced the first Korean file virus – Jerusalem.November_30th.2000.

A State of Temporary Lull

Between 1991 and 1993, things were relatively quiet. An average of 30 kinds of new viruses were found in the wild per year. Michelangelo, Dir-II, Green Caterpillar.1575, Maltese_Amoeba, MacGyver.2803, Tequila, Joshi and Quox became very widespread very quickly. Korean viruses were found too, but most of them were slight modifications of the existing ones which often came around more than once. For example, Dull_Boy, which had been found in January 1993 in Korea, was later found in old Russia in 1994 and in the USA in 1996.

More, More, More!

In 1994 the number of Korean in-the-wild viruses increased remarkably. Many of them were variants of the Jerusalem virus. Empire Monkey.B, AntiCMOS, Jerusalem.Curse.C (Jerusalem.1653.C), the Wanderer Series, the SysTurbo (BoxBox, UVScan, V3SCAN) series and Die_Hard_II were all found in 1994. Specifically, Monkey.B and AntiCMOS were extremely prevalent everywhere.

Korean virus writers, as represented by various groups such as SVS (Seoul Virus Society) and HWB (Hacker World BBS), were the first to introduce many new types of virus – stealth, encryption and polymorphic. Unfortunately, they were also instrumental in opening virus source codes to the public, which caused many variants to run rampant. Later, SVS changed its name to KOV (Knight of Virus) and produced the infamous Korean virus Burglar.1150.

Virus Toolkits

In total, 128 species of in-the-wild virus were found in Korea in 1995. This numerical increase resulted from the all-pervasive development of PC communications and the Internet boom. More worryingly, members of the public could connect to the Internet without any difficulty and some of them downloaded virus toolkits like PS-MPC and NRLG, and uploaded them on BBS. Inquisitive teenagers, mainly, were fast becoming virus producers.

Many viruses based on this kind of toolkit were found in the same month – August 1995. The Burglar series, the Eddy series, Natas.4744, One_Half.3544 and Tremor were just such examples. Significantly, the Burglar.1150 virus, which is often mistaken as having originated in Taiwan, was spread all over the world and caused a certain degree of damage. It was in March of that year that my company, the first organization to develop anti-virus software in Korea, was established.

The Heyday

In 1996, 225 new viruses were found here. Statistics show that 68% of them were of Korean origin, as opposed to 32% of foreign imports. Breaking down by type, 83.2% were file viruses, 12.8% were boot/file viruses, 3.5% were boot viruses and, in those days, only 0.4% were macro viruses.

Over half of the total number were variants – Eddy, SysTurbo and Scorpion among them. These had all resulted from virus sources or production kits being opened. Taek Soft distributed the Scorpion (Taek) virus series to the public and caused significant damage, disguising his creation as an MDir (the most famous DOS shell program in Korea). Incidentally, Taek Soft was to produce a *Windows 95* virus later in 1998.

Further to the above examples, BootExe.451, Exebug, Delwin and Euthanasia (Hare) were widely spread at this time. BOZA, the first *Windows 95* virus in the world, was found in Korea in February, and a *Word* macro virus made its first appearance in June of that year.

Attack of the Monster Macros

In 1997, the number of reports about new viruses increased. This time, interestingly, 66% of them were Korean-made. As to type, the file virus still made up the largest proportion of that number, while boot/file viruses decreased and boot viruses increased. The most significant and worrying statistic in that year was that macro viruses had more than doubled, making up 6.3% of the total number. Such unprecedented growth was a sign of things to come.

That year was later characterized by a full-scale increase in macro viruses. Korea is unlike other countries in that local wordprocessor software enjoys an 80% market share, so damage by *Word* macro viruses was relatively small. However, XM/Laroux, found in March of that year, spread widely and many companies encountered difficulties in their business. In addition, FCL (Level3.4910 virus), Kaczor.4444, and Spanska.4250 were prevalent too.

Strong Winds of Change

1998 saw yet another increase in the total number of viruses reported in Korea. While native Korean viruses fell in number very slightly, macro viruses flourished again. These viruses now made up 13% of the total number, nearly doubling in volume over the course of another year.

In early 1998 the first legal action was taken against virus producers. Four members of a prominent Korean group, the CVS (Corean Virus Club) were arrested. It marked a shift in attitudes and in procedure – at present, several other virus writers are also under investigation.

The CIH virus, found here in June, had wreaked havoc on its global journey so far. Unfortunately, it hit Korea very hard. Programs infected by this virus had been run by the five biggest national PC communication services. As a result, tens of thousands of people who had downloaded the programs had suffered losses. In addition to this, CIH-infected CDs had been distributed to readers of computer magazines and spread the chaos further afield. At the end of December, the Alt_X and Crow series, CIH, XM/Extras, HPS.5116 and Marburg virus were all responsible for significant damage.

The Present and The Future

At the present time, the most frequently encountered viruses in Korea are the *Excel* macro viruses, CIH, Monkey, AntiCMOS, One_Half, and Delwin. As in other far-eastern countries, the *Word* macro virus does not perform to its full potential here, but damage caused by *Excel* macro viruses is relatively big news because national corporations are usually the victims. Marburg, HPS, Padania, and Anxiety, all *Windows* viruses, have already been reported here, and the number of new *Windows* viruses is increasing. A Korean *Windows 95* virus has also been found.

Every year one or two new virus writers come onto the scene and they usually produce more than 50 viruses. It has become apparent, however, that most of them are only active for about a year or so before they disappear. With legislation becoming more widely implemented it is not unusual to hear of arrests and even imprisonments – a situation which, hopefully, may lead to a reduction in virus production in Korea.

A DAY IN THE LIFE

AVERT-ing Disaster

Vincent Gullotto
Network Associates

[*This series focuses on the daily work facing anti-virus professionals in both corporate and technical positions. This first feature is from the Manager of* McAfee Labs. *Ed.*]

At the forefront of the anti-virus industry's research is a group from *Network Associates* named AVERT (Anti-Virus Emergency Response Team). *NAI's* AVERT is located in eight countries spanning five continents, effectively covering enough time zones to manage a virus outbreak anywhere in the world, 24 hours a day, seven days a week.

Managing virus outbreaks is one of the many facets that make up this team of researchers, support and programming staff. The marketing team at *NAI* would love this piece to be dedicated to the kudos of AVERT, and to a certain extent it is. However, this work does go on elsewhere and this 'day in the life' is also about the bigger picture of how it is done and what this team does to succeed – namely networking and who it does it with.

Network you say; yes, network. Now, while most might think this term and practice is reserved for sales folks and the local Chamber of Commerce, there is a tremendous amount that goes on both inside and outside the four walls of AVERT in addition to the rest of the anti-virus research community. I will explore and discuss the networking aspect of the researcher's day both directly and indirectly through this article.

AVERT is comprised of several groups of professionals all dedicated to supporting a very large customer base. From the analysts, who deal directly with customers and new viruses first, to the advanced research groups, a great deal of daily interaction goes on within the team as well as flowing out to other research teams and companies. This information moving to and from the group is used to provide *NAI* customers, as well as our competitors/associates, with information and protection from 'The Black Hats', as one former *VB* Editor and target put it.

From all this support, research, problem-solving, and networking we can get an impression of 'A Day in the Life of a Virus Researcher'. Bear in mind that without the contributions to their work from others, they would probably have endless days.

All in a Day's Work

On any given day, an AVERT researcher may *just* read and write email to and from various sources. To begin with there are the requests for virus analysis. Typically, these come from analysts who see the samples from customers after they are scanned by AVERT's scanning and filter system NAISA. This system parses out the clearly identified, infected samples which have been submitted and a coordinator or analyst will respond to the users with the results and action they need to take.

If there is no clear indication of infection the analyst uses some additional tools, both automated and manual, to determine if there is an infection. With these results they decide to pass it to a researcher for further analysis and conclusion, or tell the customer that no infection is present.

The conclusion comes from the researcher looking through the file's executable code. Once there is confirmation of replication (it may not replicate and though they may not like it, they do look at, dare I say, 'malware') or not, and viral code, a fix must be implemented. The initial fix or detection and cleaning method is implemented into the product the customer is using. From there the researcher will need to make sure the fix goes into each of the products that *NAI* develops.

For example, there is an automated process that will get the fix to the user within the hour for *VirusScan v3*. For *VirusScan v4*, the researcher coordinates with AVERT-Aylesbury (UK), if they are not at that location. This fix is made available as an extra DRV, or DAT file to any infected customer. It will also be combined with all the code created by the entire AVERT research staff and the full signature set made available to customers on a weekly basis.

While this work makes up a good proportion of *any* researcher's day there are many more responsibilities they are accountable for. While a fix for the virus may be provided, there is always the question as to what it does? This will naturally be followed by, is there a description available for the customers and will it make a good piece to submit to, let's say, *Virus Bulletin* for the analysis column? More time is spent in that potentially endless day, as the marketing aspect kicks in, along with the need to promote the researcher's work to some extent.

It seems like there is always a test for the researcher to review and results to be discussed, whether a *VB* comparative, the Hamburg results, or just the latest, unsolicited test done by someone looking to get a foot in the anti-virus door. The questions are mulled over by all. How well did we do? How do we get even better results next time? What did our friends down the block score, and again, how can this be used to promote the work that is done?

Though there is typically a group that works with the researcher at one of the 'primary labs', there are also those 'in the field' at other sites, slightly more remote. This is where the internal networking continues. No matter where the information is, and who has it, it must get to everyone, and quickly. This will start off an email thread – especially if the virus of the moment has substance – the like of which most other departments have never seen.

Then there are the collections. There is not much to say about them really – they must be done, and they must be shared. The continuity of this process is vital to the anti-virus industry as a whole. It is also one of the backbones that ensures both protection for the customers and the continuation of future research.

And the Beat Goes On

As I mentioned at the outset, work similar to this is done elsewhere. However, some people may say that with all the resources available to *NAI* – big company, big AVERT name and plenty of staff to go around – this work must be easy; on the contrary. *NAI is* big and this makes the task all the more rigorous. The microscope is on this group every day, and thus expectations are high.

While this work does go on here and at *Kaspersky Lab*, *DataFellows*, *Sophos* etc, the point is that it is anything but glamorous and self-contained. It also means the researcher spends a great deal of his day networking outside the four walls of *NAI*, or the company he or she does this work for.

This is the kind of down-in-the-dirt, problem-fixing research that thousands of customers wait for around the world. Within the 'network' there is mention of who did what, and where, but from the outside looking in, there is nothing. Thus, in and of itself, the network is available for this group of guardians. It has many routes and contains a vast amount of information. For the researcher to be effective and make the most of the day, not to mention being good at the job, using the network is essential.

While indispensable, it is also a network of intense conversation and serious players that is not available for everyone. Sure, there are newsgroups and web sites, but the real information is found in the groups within the confines of the *clique* itself. From there the data spirals out, and this data is mission-critical for the researcher to succeed in the daily pursuit of anti-virus research.

A Lifeline in the Making

Networking has proved to be successful and seems to serve the virus researcher. On any given day, one virus researcher or all the virus researchers may be faced with a new and/or different virus that makes some type of impact.

This usually generates a great deal of networking and AVERT researchers, among others, get busy sending and receiving any and all information they can discover about it. In addition, not only do researchers network within their own groups, but they also begin the process of once again networking internally with the immediate team and those maniacs in marketing, always planning and positioning.

Paix Day

Prior to becoming the group's manager I experienced for the first time a real-life virus emergency and the subsequent networking. The virus was XF/Paix, discovered by *NAI* researcher François Paget in France. This virus was the first of its kind to use *Excel* formulas to infect, unlike the macro viruses we had come to know.

After having received a sample from a large, French, corporate customer, François confirmed his analysis with *NAI* Santa Clara AVERT, and samples were sent to the 'community' by Jimmy Kuo. Straight away, the networking began and as the email started moving along speculation mounted. The implementation of providing detection and removal of the virus for customers got under way. All our researchers were sent information and samples for analysis, as this would reconfirm all the *NAI* analysis, and present the opportunity to get as much information to those in the network as possible.

Thread after email thread was developed and answered. Opinions and assessment from everyone in the network followed – some believed this to be a serious threat and that Paix would be a virus that spread rapidly, as it was still a macro virus. There were others who had doubts and thought this was another *NAI* virus hype designed to create fear and give *NAI* the competitive edge. Telephone calls were made and received from researchers, customers, and the press. All the votes were in and in the end XF/Paix became one of the most prevalent and costly viruses of 1998.

In a high impact situation such as this, the researcher becomes the person in demand as companies prepare to provide their customers with the detection and repair necessary in case infection strikes. The network that provides information about the virus and its characteristics and can act as a source to trace the virus and its origin if no-one has taken credit for it already. It will also provide information about where it has been reported, so the groups can get the information back up through the channel to those who need to carry the information to others.

All of this, and there are still 100 'regular emails' to answer and a collection to review. On top of that, those Product Management folks have not yet received the outline to the latest technology that the application programmers will need to complete their portion of the next product.

When the Day is Done

I have seen many days like this since coming to work in AVERT. For the most part, days are filled with researchers working from sunrise to sunset – not quite all night, like in the emergency scenario we and other anti-virus companies have had. Earlier I mentioned those who support the researcher, and the significance of the researcher returning support. This work never ends and the viruses will keep coming. The networking never stops, and the issues will keep mounting – that is the nature of virus research.

TUTORIAL

Wordly Wise

Nick FitzGerald

Recently I have had to deal with many people afflicted with *Word 97* viruses. This is partly due to ColdApe and Brenda, which bring their victims to my attention, but also to other class-infectors that have become common in the last few months – particularly variants of W97M/Class and Ethan.

Several important observations arise from this experience. In this article I will share some useful strategies for dealing with this currently problematic class of viruses and reinforce some good advice which you have probably heard before, but which many fail to heed.

Introductory Class

What *is* a class-infector? A brief historical summary of the development of *Word* macro viruses should bring us to a reasonable understanding of this.

The earliest *Word 97* viruses were upconverted WordBasic viruses, created when infected *Word 95* documents were opened in *Word 97*. Only added at the last minute, *Microsoft* included some simple protections against *Word 97* upconverting the commoner WordBasic viruses to VBA form. Despite that, the upconvertor still translates many viral macros into functional VBA viral macros, much as it transforms useful WordBasic program macros into useful VBA program macros.

In general, the upconversion process translates each WordBasic macro into a separate VBA module. Further, because many WordBasic functions and statements do not have direct VBA equivalents (or require substantial rewriting to fit the VBA object model), *Word 97* provides the WordBasic property to ease the upconversion process. This is clearly seen by comparing the Tools, Macro listings of WM/Wazzu.A and its upconvert W97M/Wazzu.A. (It and the C variant were the first *Word 97* viruses reported in the top of the WildList.)

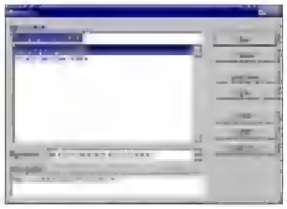


Fig. 1: The macros of WM/Wazzu.A and W97M/Wazzu.A

Now compare how this WordBasic code snippet:

```
Sub RndWord
    FileSummaryInfo .Update
    Dim dlg As DocumentStatistics
    GetCurValues dlg
```

is rendered into its VBA equivalent:

```
Private Sub RndWord()
Dim wordNum
    WordBasic.FileSummaryInfo Update:=1
    Dim dlg As Object: Set dlg = WordBasic._
        DialogRecord.DocumentStatistics(False)
    WordBasic.CurValues.DocumentStatistics dlg
```

VBA provides much richer programming capabilities than WordBasic, and the *Office 97* applications include a more comprehensive development environment. The Visual Basic Editor (VBE) is accessed through the default shortcut key Alt-F11 in the English version of *Word 97*.

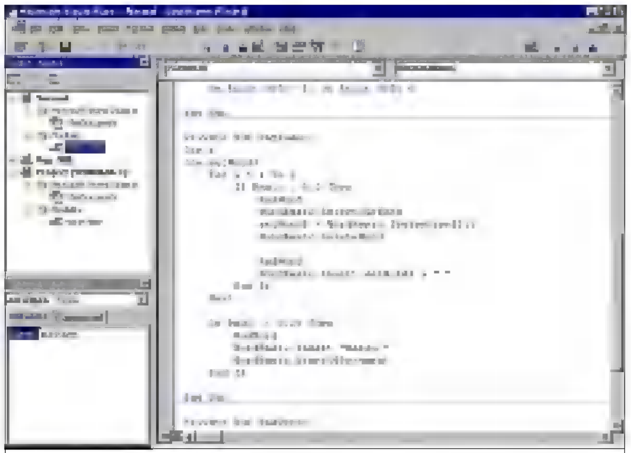


Fig. 2: The VBE of a Wazzu.A infected *Word 97*.

If you have not familiarized yourself with the VBE yet, now would be a good time to start – better that than while trying to deal with a suspicious document. Look around the various menus and toolbars. The most important part of this environment is the Project Explorer – the top left pane in the standard three-panel layout depicted in figure 2.

Project Explorer shows you at a glance where most of the objects are in a project. It is fairly clear in figure 2 that Wazzu.A has infected both the Normal project (template) and the W8WAZA-1 project (document). It appears that W97M/Wazzu.A consists of a single macro, but note the Code window shows the end of one sub-routine and two flagged as Private. This explains why the *Word 97* Macros list in figure 1 displays two autoOpen macros and not Payload and RndWord. More correctly than above, the upconversion process translates *public* WordBasic macros into separate VBA modules. Figure 3 shows Project Explorer with a more complex example – another In the Wild upconvert, W97M/Appder.A. It consists of two macros in the global template, or in VBA terms, two modules in the Normal project. In documents, it also makes a copy of the AutoClose macro to AutoOpen.

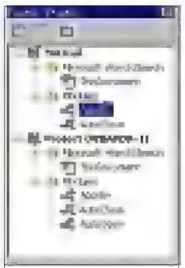


Fig. 3: Appder.A

Right-clicking a module in Project Explorer produces a context menu. If using the VBE with a suspected virus or other dubious macro, the most useful actions are likely to be Export File, which exports the selected module's source code to a text file, and Remove <module>… which deletes the named module from the project.

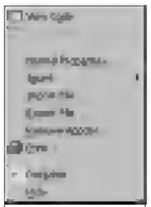


Fig. 4: Project Explorer's context menu.

Advanced Class

WordBasic macros *will* upconvert to VBA Modules, which are rather obvious in VBE's Project Explorer. Perhaps fortunately, the early days of *Word 97* virus development saw few macro virus authors exploring the riches of the VBA environment. The early, native VBA viruses simply copied the mechanisms and strategies seen in upconverted WordBasic viruses and exploited the simplicity occasioned by *Word 97* not limiting active macros to templates.

For quite some time, relatively little development was seen in Word macro viruses. That changed with the increasing deployment of the Service Release One (SR-1) version of *Office 97*. Despite the multitude of bug fixes, SR-1 might be most notable for an unannounced change – the developers crippled part of the functionality of the MacroCopy and OrganizerCopy methods of the WordBasic property.

The change did not completely disable these commands – this would have broken most legitimate macro package installers. However, these changes resulted in MacroCopy and OrganizerCopy silently failing if the destination of the copy operation was not a global template. This meant that under SR-1 versions of *Word*, most then-current, and all common, macro viruses effectively became 'intendeds' rather than real viruses. At most, they could infect global templates from documents but not spread from there to other documents. They were denied recursive replication.

This change had a major effect on some virus writers. SR-1 presented a fresh challenge – how to write an effective *Word* virus without using the stock replication mechanisms. The solution was found before many of the less-skilled virus authors were even aware there was a problem. W97M/AntiSR1.A heralded the beginning of an era of increasing complexity in macro viruses.

Having overcome the 'SR-1 problem', the Export/Import procedure became the infection method *du jour* for *Word 97* macro viruses. Apart from this, however, interest in several more esoteric pursuits seemed to grip some macro virus authors. The ensuing months saw rapid development of viruses for the 'unconquered' *Office* components (*Access* and *PowerPoint*) and VBS, cross-platform macro and VBA/VBS viruses, increased use of polymorphism and further interest in novel replication mechanisms – presumably to thwart heuristic analysers. Several of these 'advances' have fed off each other. One development amongst these was class-infection, first seen in a *Word* virus about ten months ago in W97M/Class.A.

Class objects, which contain VBA code, can exist in two places in a *Word* document. The obvious ones (if there are any) exist in the Class Modules collection, displayed in Project Explorer. You can create one from the Insert menu item (see figure 4) of Project Explorer's context menu. Selecting Class Module will produce a collection object and a class module called Class1. *Word* also allows class objects in another container – the ThisDocument object. VBA code there is not as obvious in Project Explorer as the existence of VBA Modules and Class Modules. To see such code, you must select ThisDocument then execute the ViewCode command from various places, press its shortcut key (F7) or double-click the ThisDocument object.

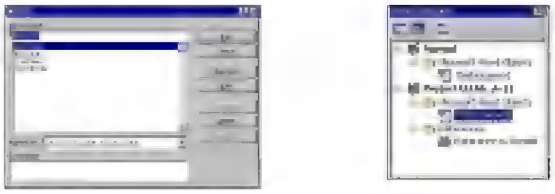


Fig. 5: The macro list and Project Explorer views of a W97M/Class.A infection.

Compare the Macro list and Project Explorer windows in figure 5 with those in figures 1 and 2. Those in figure 5 are from an environment infected with W97M/Class.A. Note that unlike with Wazzu, the location of the macros' VBA code is not obvious from Project Explorer.

Problems

Although the existence of VBA-bearing class modules had been known in anti-virus circles for some time, several products were not ready for class-infectors, insofar as they did not look for VBA code in class objects. Unfortunately, some W97M/Class variants had lucky breaks and made it into the wild before these vendors shipped versions that reliably checked class objects.

As described by Katrin Tocheva (see *VB*, March 1999, p.6), combination class infections can cause problems. Multiple infections which are viable (where both viruses keep replicating) can be missed by products which look at the whole block of code in an object, as the 'extra' code from one virus prevents reliable detection of the other. This can happen even when each virus can be separately identified.

Another problem is that when combinations of the currently popular infection methods (AddFromFile, AddFromString, InsertLines, and the perennial Export/Import) meet, they can produce invalid VBA composites (or 'sandwiches' in Tocheva's terms). The VBA environment does not seem to check code as it is inserted (replicated in the case of a virus) with these methods. However, a compiler error will be seen on closing an affected document or on exiting *Word* in the case of NORMAL.DOT.

What should you do if you encounter a document in such a state – perhaps detected by your scanner as 'probably infected with a new virus' or causing VBA errors on opening or closing. Most vendors would recommend that you send it to them and they will analyse and fix it. Whilst good advice, I have heard many tales of sensitive documents being 'mislaid' in such circumstances and sometimes you just cannot wait – there may be an unavoidable need to work with that document. Is there anything that you can do?

Declassified Documents

Despite conspiracy theories to the contrary, anti-virus vendors are not trying to maintain a closed-shop in asking that you send them such files. You may have a new virus that they have a real interest in analysing and adding to their product. They realize too that the following procedure carries some risks unless executed very carefully and thoroughly. The main concern is that incomplete removal of a virus with multiple components can result in the production of a new variant. This has already happened with the incomplete manual disinfection of O97M/Tristate.C producing a viable, *Word 97*-only virus.

This procedure should only be necessary in cases of multiple infection causing VBA compiler errors *and* when the affected files contain customizations you do not wish to lose, or for viable multiple-infections your scanner cannot clean. This is an onerous process should you have many afflicted documents, but useful for emergency cases.

This approach requires access to the VBE. Some viruses usurp the ViewVBCode command, denying access to VBE via the standard menus and Alt-F11 shortcut. To deal with this, on a clean machine make a template with this macro:

```
Sub AccessVBE()
    ShowVisualBasicEditor = True
end Sub
```

Now make a toolbar customization to run this macro. Right click a toolbar, select Customize and then click on Macros on the Commands tab. Drag this macro to the toolbar. Save the template then copy it to *Word's* Startup directory on the infected machine before starting *Word*.

Start the infected *Word* then access the VBE via the toolbar button. Explore the virus' structure, determining what to remove. Now select and delete all the subroutines belonging to the viruses present but leave any routines of your own. Next, in Project Explorer, select and delete any VBA modules that are part of the virus, again leaving your own modules. Repeat these steps for VBA code resources attached to User Forms. It is very important to be sure you have removed all the viral code at this point. Finally, save the now disinfected project.

Don't Forget…

Most successful macro viruses disable the *Word's* Macro virus protection option. Once the clean-up is done, be sure to re-enable it on all affected machines. Use a document with a harmless AutoOpen macro (MsgBox "Boo!!!" will suffice) to reinforce to affected users that they should never enable macros or customizations in unknown documents.

PRODUCT REVIEW 1

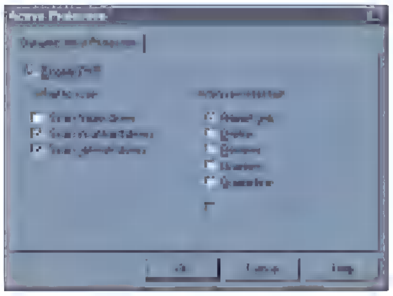
Command Antivirus for Windows NT v4.54 (SP1)

Martyn Perry

This month's review takes a look at *Command Software's AntiVirus* with *F-PROT Professional* (*CSAV*). The product's packaging comes with a splash claiming 100% 'In the Wild' virus detection and 100% 'In the Wild' virus disinfection. With new viruses appearing daily, how well do these bold claims stand up to scrutiny?

Presentation and Installation

The product ships on CD and comes boxed with a multi-platform Quick Start Guide. The rest of the documentation is stored in *Adobe Acrobat* (PDF) format. The documents include the administrator's manual as well as Joe Well's virus encyclopaedia.



The CD also contains various folders for the *NT* and *NetWare* server versions. The CD does not auto-start, thereby giving the opportunity to review what is available. To install, go to *NT Server* directory and call Setup or run setup netadmin. Initially, an operating system check is performed and if the machine is not *NT Server*, it will issue a warning and exit – installation cannot be performed from a remote workstation.

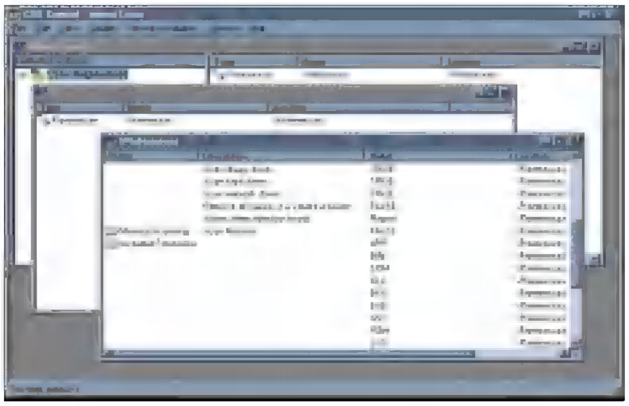
The installation default path is C:\Program Files\Command Software\F-PROTNT. There is a default setup which takes the default settings and also does not install CSS Central (the administration module) or there is the custom option. The default option was tried first.

This option asks if scheduled scanning of network drives is required. If it is, then it is necessary to ensure that CSS AV Scheduler service is started with an existing account that has rights to access network drives. If this is not available then scheduled scans are only provided for local drives.

If scheduling is deselected then installation proceeds by copying the files. If scheduling is selected, there is a request for a suitable account. The user is prompted for a domain, username and a password which will not expire. There is a check for previous versions and then the files are copied. Next comes a request whether to create a rescue disk. The decision taken in this instance was 'no'. The installation now completes and displays a readme file.

The custom option gives the choice of selecting README files, HELP files, CSS Communication agent and CSS Central (both used for managing machine groups.) Three choices of default directories are offered – CSS Central Directory C:\... \CSS Central, Destination Directory C:\... FPROTNT and quarantine directory C:\Quarantine.

The program is started by calling F-PROT32.EXE or from the task bar by a right click on the C icon (F-agent). F-agent also provides access to the *NT* viewer as well as displaying the statistics dialog box. If the F-agent is closed, then inactivity scans will not run and real-time scans will not display any user notification.



Scanning Options

Scanning can be tailored to all files or specific extensions. The default list is the same as for Manual and Scheduled scans. In the event of a virus being found, then one of the following actions can be chosen to deal with the situation – Report only, Delete, Rename, Disinfect, or Quarantine. With the last option, the files are moved to the quarantine directory (C:\QUARANTINE). They are renamed and a log of their source directory and file name are stored in a HISTORY.LOG file also in the quarantine directory.

The Real-Time scan or DVP (Dynamic Virus Protection) uses three separate kernel-mode drivers. One handles file systems, one filters file events and the third is the scanning engine itself.

The default set of files included in the scan are APP, BIN, COM, DLL, DO?, EXE, MDB, OV?, PGM, RTF, SCR, SYS, and XL? In a separate table is the option to create an exclusion list of files. On infection, the actions to be taken can be either Report only or a choice of Disinfect with Query first, Delete with Query first, Rename with Query first, or Quarantine with Query first.

The scheduled scan can be configured to run Daily, Weekly or Monthly. The time of day, the day of the week and the date in the month can be set as appropriate. There is an additional option to scan after so many minutes of inactivity. The default file selections are the same as for Manual scanning. For scheduled scan statistics to be updated, Dynamic Virus Protection must be enabled.

Administration

Although the product is essentially a *Windows*-based application, it is good to see that the command line options have been retained. This gives the flexibility to control the scanner operation via dial-up control if necessary. For the GUI-based version, the main screen provides three sets of menus – Task, View and Preferences.

All the scanning operations and configuration choices are managed by creating Tasks. Task options provide the ability to create, edit, or delete a task as well as to amend the properties of the task. These include settings for drive and path selection in addition to the action to take in the event of a virus being detected.

The scan method is greyed out but displayed as Secure. This harks back to the old option in *F-PROT* to select the level of scan required. Under Preferences a range of choices is available – Network messaging, Reporting options, Active Protect selections, File selection, and Advanced (meaning Administrative) options.

In the event of a virus being detected, *CSAV* provides notification either by sending a message or an email. The email can either consist of just a report of the problem or the infected files themselves. The selections for reporting are as follows: beep when virus found, list all files scanned or wrap text in report window.

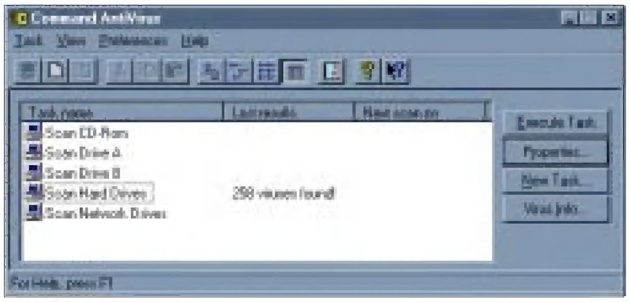
In addition to the standard administration functions, one of the product's features is CSS Central. This provides system administrators with a facility to distribute, update and modify *CSAV* setups from one location. *Command* has taken a step further in helping to distribute software by creating a package distribution file (PDF) to work with *Microsoft's System Management Server* (*SMS*).

Updates

The tests were performed using the Service Pack 1 updated v4.54 scanning engine. The definition files, MACRO.DEF and SIGN.DEF, were both dated February 1999 and were updated by simply replacing the exisiting copies with the latest versions. Automatic updates of a workstation can be performed provided that *F-Agent* is active on it.

Scanning Overhead

To measure the extra work performed in detecting a virus, a diskette comprising 26 EXE and 17 COM files was scanned. An overhead of 7.2% was measured for repeating the scan with all the files infected with Natas.4744 virus. It took 25 minutes and 41 seconds to scan 5,500 clean files. No false positives were returned, although it did report two files as suspicious – it thought SETDAY.COM could be a destructive program and DIAL.EXE could be infected with an unknown virus. In addition, it reported SHAPES.COM as a potentially corrupted COM file. Eleven files were not scanned, reporting files with SPD extensions as an unsupported compression method.



Detection Rates

The scanner was checked using the standard *Virus Bulletin* test sets – ItW, Standard, Polymorphic, Macro and Boot Sector. Tests were conducted using the default scanner file extensions supplied. The scan action option was selected to delete the infected files and the residual file count used in determining the detection rate.

All the In the Wild boot sector viruses were detected. Unfortunately however the VxD sample of Win95/Fono was missed in the In the Wild file test-set, and so the 100% claims made on the packaging were not to be realized. Standard test-set results showed that one sample of Joker and the nine Navrhar VxDs were missed. *CSAV* detected all the polymorphic samples but missed 38 samples against the Macro test-set. Identical detection rates were observed upon rerunning the tests with All files selected, the VxD sample of Win95/Fono still proving elusive and depriving *CSAV* of a 100% In the Wild detection rate.

Real-time Scanning Overhead

To determine the impact of the scanner on the workstation when it is running, the following test was executed. 200 EXE and COM files of 21 MB were copied from one folder to another using XCOPY. The folders used for the source and target were excluded from the scan – avoiding the risk of a file being scanned while waiting to be copied.

The default setting of Maximum Boost for Foreground Application was used in all cases. Due to the different processes which occur within the server, the time tests were run ten times for each setting and an average taken. The tests were as follows:

- Program not loaded: establishes the baseline time for copying files on the server.
- Program installed, but with Real-time disabled: tests the impact of the application in a quiescent state.
- Program loaded with Real-time (DVP) enabled: tests the impact of the scan on when files are accessed.
- Program loaded with Real-time (DVP) enabled, plus Manual scan: tests the full impact of running the scan for files access and the normal scanning of files.
- Program unloaded: run after the server tests to check how well the server is returned to its former state.

The effect of the DVP (Dynamic Virus Protection) has a significant impact on the performance of the virus scanner. Adding full scanning to the load lifts the overhead further as would be expected.

Summary

The bold claims of 100% 'In The Wild' detection and disinfection were almost, but not quite, realized. Should we be surprised? I think not. Detection is a moving target and the WildList is in a constant state of flux.

The list used in this test was from December 1998 with virus definitions from the beginning of February 1999. Even with all files selected, one of the samples in the In the Wild test-set and 48 samples in the other test-sets were still missed. While being critical of overzealous marketing hype, the general detection level is certainly respectable. The scan speed feels a little sluggish particularly with floppy detection. This appears to be due to the amount of time spent doing screen updates.

Command Antivirus for Windows NT

Detection Results

Test-set[1]	Viruses Detected	Score
In the Wild Boot	84/84	100.0%
In the Wild File	856/857	99.9%
Standard	1045/1055	99.1%
Polymorphic	14444/14444	100.0%
Macro	2631/2669	98.6%

Overhead of On-access Scanning:

The tests show the time (in seconds) taken to copy 200 COM and EXE files (21 MB). Each test was repeated ten times, and an average taken.

	Time	Overhead
Not loaded	14.0	–
Loaded, disabled	14.4	2.9%
— + DVP enabled	127.6	811.4%
— + — + manual scan	184.1	1215.0%
Program unloaded	130.6	832.9%

Technical Details

Product: *Command Antivirus for Windows NT.*

Developer: *Command Software Systems*, 1061 East Indiantown Road, Suite 500, Jupiter, Florida 33477-5143, USA; Tel +1 561 575 3200, email sales@commandcom.com, WWW http://www.commandcom.com/.

Price: $295 per server, includes 1 years support and updates and 24-hour toll free technical support.

Hardware Used: Workstation: *Compaq* Prolinea 590, 80 MB of RAM, 2 GB hard disk, running *NT Server v4.0 (SP3).*

[1]**Virus Test-sets:** Complete listings of the test-sets used are at http://www.virusbtn.com/Comparatives/DOS/199901/test_sets.html.

PRODUCT REVIEW 2

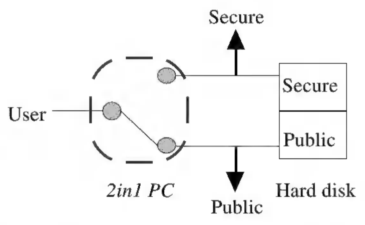
2in1 PC

Following a recent review of *Hardwall* from *Calluna Technology* (*VB,* February 1999, p.17), we continue the theme of hardware-based security methods in this test of *2in1 PC* from *Voltaire*. Would the 'bullet-proof' claims made in the product's documentation be realized?

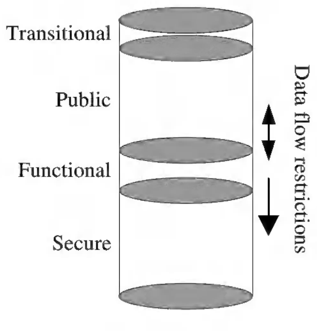
Principles of Operation

2in1 PC provides data security by dividing its host computer into two separate machines (termed *public* and *secure* machines throughout this review). This is achieved by partitioning the hard drive of the host PC, creating *secure* and *public* partitions.

Once installed, *2in1 PC* configures the computer such that the user is restricted to reading/writing data from/to only one of these partitions at any one time. In this way, two 'machines' are created, each with its own operating system, network functionality and 'virtual' hard disk.



In addition to the public and secure partitions, installation of *2in1 PC* also creates transitional and functional partitions. The transitional partition is accessed when the PC is booted, or in switching between the public and secure machines. The functional partition is (optionally) created in order to provide a facility for users to transfer data between the public and secure machines, and can be configured according to the needs of the user. Typically, a configuration would be used such that the leakage of data from the secure partition was prevented.



The *2in1 PC* has been designed to cater for users who require network access. In an environment where the user is connected to both internal (e.g. LAN) and external (e.g. Internet) networks, *2in1 PC* controls access to those networks according to whether the secure or public domain of the hard drive is active respectively. Thus, important files stored on the secure hard disk partition are not accessible whilst the user is browsing the Internet. Furthermore, any files downloaded or installed onto the public partition can only be transferred via the functional partition.

In order to prevent a security breach between the secure and public machines, the *2in1 PC* manages the master boot record (MBR). Any attempts to read the MBR are supplied from the EEPROM rather than the hard disk.

The Package

The product arrived in a small box decorated with the *2in1 PC* banners and the logo of its manufacturer, *Voltaire Advanced Data Security*. A few bold statements concerning the effectiveness of the product are presented, including 'Simply keeps your data secure' and 'Two physically separated network connections in a single PC'. The final claim 'Voltaire guarantees your data security by separating a single PC into two over time' provided the real incentive to tear the box open and install the review product.

Contained in the box are full and quick installation guides, two 3.5-inch installation diskettes, a variety of IDE and network cables and the *2in1 PC* card itself. Despite being packaged in bubble wrap, the card was free to rattle about within the box which perhaps explains some of the problems that were encountered during testing.

The full installation guide is 81 pages long in all, divided into ten chapters. The bulk of the manual is concerned with the installation and configuration of *2in1 PC*.

Installation

The specifications of the PC used for testing are given in the technical details box at the end of this review. Prior to installation, SCANDISK and DEFRAG routines were run as recommended in the installation guide. For users who are unfamiliar with these disk maintenance programs, step-by-step instructions are provided for *Windows 3.1x*, *95*, *98* and *NT* operating systems.

It became clear early on in the installation process that a potentially off-putting feature of *2in1 PC* is that only one existing hard drive partition (of at least 500 MB) is supported. Given that *PartitionMagic* from *PowerQuest* is supplied with the product, it is perhaps surprising that the installation process does not guide the user through the removal of any existing partitions. In addition to the removal of multiple hard disk partitions, installation of *2in1 PC* requires that any existing software that records information from the MBR (eg some anti-virus or encryption programs) must also be removed to avoid malfunctions during the installation process.

Physical installation of the card is fairly straightforward. The supplied IDE cables are used to connect the hard drive controller to the card, and the card to the primary IDE socket on the motherboard. Once connected, the card is inserted into a vacant 8- or 16-bit ISA slot.

Configuring 2in1 PC

In order to alter the configuration of *2in1 PC*, the set-up plug must be inserted onto the card (enabling write-access to the EEPROM chip on the card). Once configured, this plug should be removed, thus preventing unauthorized alterations to the machine configuration. The configuration program is accessed by rebooting from the first of the two installation diskettes.

A diagnostics routine initially runs to verify that the system is correctly set up for installion of *2in1 PC*. At this point the reviewer's patience was sorely tested by the repeated occurrence of a system error due to the hard disk not being detected, which caused abortion of the installation/configuration program. A second card was sent for, received and tested. With this card the hard disk was detected and the configuration program loaded successfully.

During installation (and subsequent reconfiguration) data is written to the first installation diskette explaining why the disk was received in a write-enabled state. Once write-access is confirmed the user is prompted to choose either a default or custom installation method. The default option suggests a suitable partition scheme based on the overall size of the hard disk. An alternative scheme can be adopted by choosing the custom installation option. Once the desired partition sizes are settled, *PartitonMagic* is executed and automated drive partitioning proceeds. Once finished, a summary of the new system configuration is displayed and the user is prompted to reboot into the secure partition to install the *2in1 PC* drivers.

Finally, installation of an operating system and drivers into the public partition is required to complete the installation process. The importance of previously creating a boot disk complete with CD-ROM device drivers should be emphasized in the installation guide. Manual editing of the CONFIG.SYS and AUTOEXEC.BAT start-up files was required in order to enable the CD-ROM such that any software could be installed from a CD.

Though not a complex procedure, it is a tiresome one which is perhaps forgotten in this world of *Windows*-based virtual device drivers. Having enabled the CD-ROM no further problems were encountered during installation of various operating systems into the public partition and the *2in1 PC* drivers were subsequently successfully installed.

Working with 2in1 PC

It is possible to configure the PC to boot into either the public or secure machine by default, or a menu can be presented to the user offering the choice. Once in one of the machines, switching to the other is achieved by double-clicking an icon on the Windows desktop (and in the taskbar for *Windows 95/98/NT* installations). This activates a switching cycle where the PC is booted twice, firstly into the transition partition from which customized software can run, and then into the desired secure or public partition.

Access to the functional hard disk partition from the public and secure machines can be individually controlled from within the set-up mode of the *2in1 PC*. Access can be disabled completely, set to read only, or set to read and write enabled. The most logical set-up would typically be for users on the public machine to have full read/write access, but for users on the secure machine to be restricted to read only (see above figure). Thus data can be transferred from the public to the secure machine via the functional partition, but no data can flow the opposite way. All permutations of the functional drive access settings were tested, and the card perfomed as claimed, restricting access to the partition according to the *2in1 PC* configuration.

The provision of a facility to allow data transfer between the public and secure partitions may appear to be a hole in the security envelope provided by *2in1 PC*. Software can, however, be installed into the transitional partition such that it is activated during switching between the public and secure machines (and on machine bootup).

As an example of this, by installing anti-virus software into the transitional partition and editing the batch files that enable booting into either the public or secure machines, it is possible to force virus scanning of all files on the functional partition. Alternatively, access-control software could be installed, enabling password controlled access to the secure machine.

The use of *2in1 PC* in a multiple network environment was tested. The product worked as claimed, changing the network connections upon switching between the secure and public machines, such that the two networks were completely isolated from each other with only one being accessible at any one time.

Conclusions

2in1 PC is not an anti-virus product. It is a security product designed for the separation of data on a single workstation. To that end it worked as claimed, creating two separate, virtual machines where access is controlled by the administrator. The installation of the card was reasonably straightforward if a little messy, particularly when installing software into the transitional partition. However, the main drawback of *2in1 PC* has to be its price.

Technical Details

Product: *2in1 PC.*

Developer: *Voltaire Advanced Data Security Ltd,* 103 Medinat Hayehudim, POB 12534, Herzelia 46733, Israel; Tel +972 9 9512177, email info@voltaire.co.il, WWW http://www.voltaire.co.il/.

UK & Australasia Distributor: *Portcullis Computer Security Limited*, The Grange Barn, Pikes End, Pinner, Middlesex HA5 2EX; Tel +44 181 8680098, fax +44 181 8680017, WWW http://www.portcullis-security.com/.

Price: £268 per card – discount for bulk orders.

Hardware Used: 166MHz Pentium-MMX with 64 MB of RAM, 4 GB hard disk, CD-ROM drive and 3.5-inch floppy. Configured to run *Windows 95/98/NT* and DOS 6.22.

SUBSCRIPTION RATES

Subscription price for 1 year (12 issues) including first-class/airmail delivery:

UK £195, Europe £225, International £245 (US$395)

Editorial enquiries, subscription enquiries, orders and payments:

Virus Bulletin Ltd, The Pentagon, Abingdon Science Park, Abingdon, Oxfordshire, OX14 3YP, England

Tel 01235 555139, International Tel +44 1235 555139
Fax 01235 531889, International Fax +44 1235 531889
Email: editorial@virusbtn.com
World Wide Web: http://www.virusbtn.com/

US subscriptions only:

Virus Bulletin, 18 Commerce Way, Woburn, MA 01801, USA

Tel (781) 9377768, Fax (781) 9320251

END NOTES AND NEWS

Full details of the upcoming VB'99 conference can be found at http://www.virusbtn.com/. The call for abstracts is now closed and a full brochure will be available soon. The 9th annual *Virus Bulletin* conference will run from Thursday 30 September to Friday 1 October at the Hotel Vancouver, Vancouver, British Columbia. Contact conference co-ordinator Joanne Peck for details about exhibition space; Tel +44 1235 555139, fax +44 1235 531889, or email Joanne.Peck@virusbtn.com/.

***Peapod Ltd* announce the release of *Trend Micro's ScanMail for Lotus Notes v1.8* for *Solaris* and *AIX*.** The product provides real-time scanning at all three of the main entry points for virus infections – Database modifications, *Notes* mail attachments and replicated documents. *Trend* claims that *ScanMail* protects against HTM and HTML viruses that can be embedded in Web pages or electronic mail messages. It has also been upgraded to the high-speed *VSAPI 2.0* scan engine. For further information contact Steve White at *Peapod Ltd*; Tel +44 181 6069990 or email trend@peapod.co.uk.

***Data Fellows* has added the detection of NetBus 2.0 Pro to *F-Secure Anti-Virus*.** NetBus is a remote administration tool for *Windows* similar to the infamous Back Orifice tool. This means a *Windows* workstation can be accessed using the NetBus client, allowing a machine to be controlled across the Internet, even from another country. Since February 1999, NetBus has been marketed by its developers and enhanced with new features. Older, free versions have been detected by most anti-virus products. Further to several requests from corporate clients, *Data Fellows* has decided to make this an optional feature as NetBus is widely used in Northern Europe. *F-Secure Anti-Virus* detects NetBus 2.0 Pro as 'Backdoor.NetBus.20'. An update to include detection of NetBus is available at the company web site http://www.DataFellows.com/. For further details about this addition, contact the *Data Fellows* offices in San José, California; Tel +1 408 9386700, fax +1 408 9386701.

***Sophos* will be hosting an introductory computer virus workshop on 19 May 1999 to be followed on 20 May by an advanced session.** The two-day course will be held at the organization's training suite in Abingdon, UK. To register for a place on the course, contact Karen Richardson; Tel +44 1235 544015, fax +44 1235 559935, or find more information at http://www.sophos.com/.

Infosecurity'99 is to be held at the National Hall in Olympia, London from 27–29 April 1999. For more information contact Richard Harris; Tel +44 181 9107718 or visit the conference web site http://www.infosec.co.uk/.

Russian anti-virus software developer *DialogueScience* announces the inclusion of a new program in its latest release of *Dr Web* for Win32. Spider is a resident anti-virus program for *Windows 95/98*. Further details about this and the latest version numbers can be found on the company web site at http://www.dials.ru/.

***Symantec* announced the launch of *Norton CleanSweep v4.5* in mid-February 1999.** The company claims that this new version is the quickest and easiest way to recover hard disk space safely. The *Norton CleanSweep Deluxe* version includes a free CD copy of Zip-It, a free six month subscription to *Norton* Web services and an offer from Netcom of 150 free hours of Internet access for users who subscribe to the service. *Norton CleanSweep v4.5* is available now for £39, while the *Deluxe* version retails at £49. Contact *Symantec Customer Services*; Tel +44 171 6165600 or see http://www.symantec.com/.

A call for papers has gone out for the fifteenth Annual Computer Security Applications (ACSA) Conference, to be held at the Radisson Resort, Scottsdale, Arizona from 6–10 December 1999. Suggested topics range from Internet technologies and electronic commerce to software safety and legal and ethical issues. All abstracts must be submitted by 28 May 1999. More information about the different streams and formats can be found at http://www.acsac.org/.

The *Secure Computing* Awards are to be held at the Park Lane Hotel, London on the evening of 27 April 1999. Academy awards are voted by a team of IT professionals, while the Reader Trust awards are voted for by software users and readers of the magazine. Both categories include Best Anti-Virus Solution. For additional information about the event and the awards visit the *Secure Computing* web site; http://www.westcoast.com/, or email Debbie Evans; devans@westcoast.com.

A workshop entitled 'Countering Cyber-Terrorism' is to be held in California from 22–23 June 1999. The Information Sciences Institute of the University of California is to sponsor the event. For more details see http://www.isi.edu/cctws/.